

### REMARKS

Claims 1-13, 15-20, 24-28, and 32-40 stand rejected. Claims 1, 20 and 28 have been amended to clarify the features of the invention. As a result, claims 1-13, 15-20, 24-28, and 32-40 are pending for examination with claims 1, 20 and 28 being independent claims. The amendments made find support in the specification and do not constitute new matter.

Claim 1 stands rejected under 35 U.S.C. §103(a) as being unpatentable over Baisley et al. U.S. Patent 6,330,569 B1 ("Baisley") in view of Snodgrass et al. U.S. 6,185,556 B1 ("Snodgrass"). The Examiner states that "it would have been obvious to one of ordinary skill in the art at the time the invention was made to combine Baisely and Snodgrass because, being able to change or replace programs based on criteria would make editing or updating instructions more efficient."

Applicants have amended Claim 1 to call for:

"a program matching criteria matching at least one existing executable program to be updated by performing a first partial name search within an index followed by a second whole name search within the set of program entries specifying correction information" (underlining added for emphasis)

Applicants submit that the invention as claimed in Claim1 is neither taught, described nor suggested in Baisley even in view of Snodgrass.

The present invention provides: "Next, during step 406 the Get Match API 206 searches the index 204 for the name, or portion of the name, of the executable program provided during step 400. By way of example, the Get Match API 206 compares the first eight characters of the provided executable program name to each entry within the index 204. The index 204 typically only occupies around 10 kilobytes of memory space, and is located at the beginning of the database, and therefore it can be read in one read operation. At step 408, if no matches are found in the index 204, then control passes to step 410 wherein the Get Match API 206 returns a message to the calling process that no matches were found (i.e., this program does not have a corresponding entry in the modification specifications 200). Control then passes to the End.

If however during step 408 at least one matching index entry is located by the Get Match API 206 within the index 204, then control passes to step 412. During step 412, the Get Match API 206 completes a second tier search for potential matching executable (i.e., <EXE> tagged) entries within the modification specifications 200. During step 412 the Get Match API 206 compares the name of the executable program (provided during step 400) to whole names of programs stored within particular individual EXE tagged entries of the modification specifications 200." (page 23 lines 1-24)

...

"If however during step 414 one or more whole names match the file name, then control passes to step 418. During step 418, the Get Match API 206 performs a matching operation for a first potential matching entry rendered during step 412. A criteria listed for a first potential matching entry in the database is checked against the executable program and files located in a

specified directory and/or subdirectory. The matching operation is dictated by the matching criteria specified by the potential matching entry. An entry's criteria can include, but is not limited to: a file name, a file size, a file checksum, file version information, and file creation time." (page 23, line29– page 24, line 8) (underlining added for emphasis)

Baisley, on the other hand, provides:

"11. In a computer system executing a repository program and having a memory, a method for versioning a UML model in said repository in accordance with an updated XML representation of said UML model, said method comprising the steps of:

a. traversing depth-first an object tree of said XML representation, and for each object found in said object tree:

1) matching attributes to corresponding attributes for a corresponding object of said UML model, and if they match;

2) matching references with references for a corresponding object of said UML model, and if they match;

3) retrieving all objects owned by each object of said XML representation at current depth;

4) determining if corresponding object of said UML model has the same number of owned objects, and if so;

5) determining if all owned objects match, and if so;

b. determining if traversal of said object tree is complete, and if not; and,

c. repeating all of the steps above until said traversal is complete.”. (Claim 11) (underlining added for emphasis)

...

Accordingly, the Applicants submit that Claim 1 is not unpatentable over Baisley in view of Snodgrass.

Claims 2–13 and 15–19 are dependent on Claim 1; as such, these dependent claims are believed allowable based upon Claim 1.

Claims 20 and 28 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Baisley et al. U.S. Patent 6,330,569 B1 (“Baisley”) in view of Snodgrass et al. U.S. 6,185,556 B1 (“Snodgrass”). The Examiner states that “it would have been obvious to one of ordinary skill in the art at the time the invention was made to combine Baisely and Snodgrass because, building further searches or queries when executed “eliminate duplicates”.”

Applicants have amended Claim 20 to call for:

“first at program loading for execution executing a first search on a name portion of a whole name of an index having identifying information for each one of the set of program entries to identify a first set of potential matching entries;

second at program loading for execution executing a second search on at least a portion of the first set of potential matching entries to identify a second set of potential matching entries matching the particular executable program based upon a whole name for the program entry; and

third at program loading for execution executing a third search on at least a portion of the second set of potential matching entries to identify a program entry

matching the particular program based upon the program matching criteria for the program entry” (underlining added for emphasis)

Applicants have amended Claim 28 to call for:

“first at program loading for execution executing a first search on a name portion of an index having identifying information for each one of the set of program entries to identify a first set of potential matching entries;

second at program loading for execution executing a second search on at least a portion of the first set of potential matching entries to identify a second set of potential matching entries matching the particular executable program based upon a whole name for the program entry; and

third at program loading for execution executing a third search on at least a portion of the second set of potential matching entries to identify a program entry matching the particular program based upon the program matching criteria for the program entry” (underlining added for emphasis)

Applicants submit that the invention as claimed in Claims 20 and 28 are neither taught, described nor suggested in Baisley even in view of Snodgrass.

The present invention provides:

"The present invention relates to program modifications that are dynamically incorporated into a computer program (i.e., at the time an operating system loads a program for execution)" (Area of the invention, page 1), and

"Next, during step 406 the Get Match API 206 searches the index 204 for the name, or portion of the name, of the executable program provided during step 400. By way of example, the Get Match API 206 compares the first eight characters of the provided executable program name to each entry within the index 204. The index 204 typically only occupies around 10 kilobytes of memory space, and is located at the beginning of the database, and therefore it can be read in one read operation. At step 408, if no matches are found in the index 204, then control passes to step 410 wherein the Get Match API 206 returns a message to the calling process that no matches were found (i.e., this program does not have a corresponding entry in the modification specifications 200). Control then passes to the End.

If however during step 408 at least one matching index entry is located by the Get Match API 206 within the index 204, then control passes to step 412. During step 412, the Get Match API 206 completes a second tier search for potential matching executable (i.e., <EXE> tagged) entries within the modification specifications 200. During step 412 the Get Match API 206 compares the name of the executable program (provided during step 400) to whole names of programs stored within particular individual EXE tagged entries of the modification specifications 200." (page 23 lines 1-24) (underlining added for emphasis)

...

"If however during step 414 one or more whole names match the file name, then control passes to step 418. During step 418, the Get Match API 206 performs a matching operation for a first potential matching entry rendered during step 412. A criteria listed for a first potential matching entry in the database is checked against the executable program and files located in a specified directory and/or subdirectory. The matching operation is dictated by the matching criteria specified by the potential matching entry. An entry's criteria can include, but is not limited to: a file name, a file size, a file checksum, file version information, and file creation time." (page 23, line29– page 24, line 8) (underlining added for emphasis)

Snodgrass, on the other hand, provides: "One or more queries are built 314 that, when executed by a database product optionally eliminate duplicates from the table resulting from the execution of the query built in step 312 as described above with reference to the where table optimizer 222 of FIG. 2 and/or build a constant set from the table resulting from step 312 or 314 as described above with reference to the where constant set builder 224 of FIG. 2." (see col. 21, lines 33–40; underlining added for emphasis)

Accordingly, the Applicants submit that Claims 20 and 28 are not unpatentable over Baisley in view of Snodgrass.

Claims 24–27 are dependent on Claim 20; as such, these dependent claims are believed allowable based upon Claim 20.

Claims 32–37 are dependent on Claim 28; as such, these dependent claims are believed allowable based upon Claim 28.

#### CONCLUSION

Accordingly, in view of the above amendment and remarks it is submitted that the claims are patentably distinct over the prior art and that all the rejections to the claims have been overcome. Reconsideration and reexamination of the above Application is requested. Based on the foregoing, Applicant respectfully requests that pending claims be allowed, and that a timely Notice of Allowance be issued in this case. If the Examiner believes, after this amendment, that the application is not in condition for allowance, the Examiner is requested to call the Applicant's attorney at the telephone number listed below.

If this response is not considered timely filed and if a request for an extension of time is otherwise absent, Applicant hereby requests any necessary extension of time. If there are any fees due in connection with the filing of this amendment, please charge the fees to our Deposit Account No. 50-0463.

In the event that there are any outstanding matters remaining in the filing of this amendment, the Examiner is invited to contact the undersigned to discuss this application.

Respectfully submitted,



Date: October 5, 2005  
Microsoft Corporation  
One Microsoft Way  
Redmond WA 98052-6399

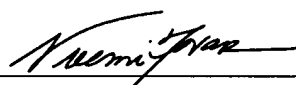
By: \_\_\_\_\_  
Paul B. Heynssens  
Registration Number: 47,648  
Direct telephone: (425) 707-3913

**CERTIFICATE OF MAILING OR TRANSMISSION [37 CFR 1.8(a)]**

I hereby certify that this correspondence is being:

☒ deposited with the United States Postal Service on the date shown below with sufficient postage as first class mail in an envelope addressed to: Mail Stop Amendment, Commissioner for Patents, P. O. Box 1450, Alexandria, VA 22313-1450

October 5, 2005  
Date

  
\_\_\_\_\_  
Signature

Noemi Tovar  
Type or Print Name